UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/602,515	06/23/2000	Leonard J. Testa	4675-008	9084

4678 7590 04/13/2004

MACCORD MASON PLLC
300 N. GREENE STREET, SUITE 1600
P. O. BOX 2974
GREENSBORO, NC 27402

EXAMINER	
VAN DOREN, BETH	

ART UNIT	PAPER NUMBER
3623	

DATE MAILED: 04/13/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/602,515	TESTA, LEONARD J.	
	Examiner	Art Unit	
	Beth Van Doren	3623	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *24 December 2003*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-54* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-54* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. The following is a Final Office Action in response to communications filed 12/24/03. Claims 1, 11, 26-28, 37, and 52-54 have been amended. Claims 1-54 are now pending in this application.

Response to Amendment

2. Applicant's amendments to claims 11, 26, and 28 are sufficient to overcome the claim objections set forth in the previous office action.

3. Applicant's amendments to claim 1 is sufficient to overcome the 35 USC § 112, second paragraph, rejection set forth in the previous office action.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Syswerda (U.S. 5,319,781).

5. As per claim 1, Syswerda teaches a scheduling system for scheduling a plurality of time-dependent tasks, said scheduling system comprising:

(a) an enumerative brute force module (See figure 5, column 2, lines 39-40, and column 5, lines 62-67, wherein tasks are listed in random order using no intelligence or reasoning);

(b) a deterministic module that considers constraints when ordering the tasks (See figures 5 and 7, column 2, lines 40-45, column 3, lines 20-40, column 4, lines 20-36, column 6, lines 1-10, and column 8, lines 1-7, that considers the constraints to determine an ordering of the tasks);

(c) a genetic module (See figure 5, column 3, lines 50-67, column 4, lines 50-67, column 6, lines 14-35, wherein a genetic algorithm is implemented); and

(d) a partitioner module for selecting one of said brute force module, said dynamic programming module, or said genetic module to generate a schedule for selecting said plurality of time-dependent tasks (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36, column 5, lines 62-67, column 6, lines 1-10 and 14-35, column 8, lines 1-7, and the appendix, which discloses a computer implementation which gives a portion ability to divide the scheduling responsibilities by selecting one of the modules. This portion has control over the flow of the schedule generations).

However, Syswerda does not expressly disclose a dynamic programming module or a hashing function having a low probability of collisions.

Syswerda discloses a deterministically programmed schedule builder that obtains a list of tasks and orders them using hard and soft constraints. Syswerda et al. also discloses when discussing the deterministically programmed schedule builder that this module can include a more complex scheduling techniques, as stated in column 6, lines 5-11, and column 7, lines 29-37. Dynamic programming considers changeable competing or conflicting forces, such as resource and time constraints, to produce multiple possible outcomes and therefore would be considered a specific type of deterministic scheduling by one of ordinary-skill in the art.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to use dynamic programming techniques in the deterministic module of Syswerda in order to increase the ability of the system to build better schedules while still acting deterministically, as stated in column 7, lines 29-40.

Furthermore, Syswerda teaches grouping schedules by score and deleting those schedules grouped at specific score thresholds as well as deterministically building schedules by applying soft and hard constraints to lists of tasks. Hashing functions are well known in the computer programming art and are used to identify and map values to a location for faster data retrieval. It is also known in the art of programming that a hash function should be fast and it should cause as few collisions as possible so that it does not produce the same value from two different inputs. It would have been obvious to one of ordinary skill in the art at the time of the invention to use a hashing function to group the schedules in order to more efficiently map schedules of duplicate task lists and/or duplicate scores to the same location for easy deletion.

6. As per claim 2, Syswerda discloses a scheduling system further comprising a constraints file, said constraints file providing an input to said partitioner module (See column 3, lines 20-40, and appendix, which discloses storing constraints in the computer implemented scheduling system).

7. As per claim 3, Syswerda teaches a scheduling system wherein said constraints file includes at least one predetermined time constraint (See column 3, lines 20-40, column 4, lines 20-40, column 6, lines 1-10, and appendix, which discloses a predetermined time constraint).

8. As per claim 4, Syswerda teaches a scheduling system wherein said predetermined time constraint further includes at least one constraint selected from the group consisting of:

(a) a precedence constraint, wherein said precedence constraint limits scheduling of said tasks according to a predetermined order of occurrence in time (See column 3, lines 20-40, column 4, lines 20-40, column 6, lines 1-10, which includes at least one of a precedence constraint, a time window constraint, a priority constraint, or a conditional constraint);

(b) a time window constraint, wherein said time window constraint limits scheduling of said tasks within a time window of a predetermined length (See column 3, lines 20-40, column 4, lines 20-40, column 6, lines 1-10, which includes at least one of a precedence constraint, a time window constraint, a priority constraint, or a conditional constraint);

(c) a priority constraint, wherein said priority constraint limits scheduling of said tasks to a sequence according to a predetermined priority (See column 3, lines 20-40, column 4, lines 20-40, column 6, lines 1-10, which includes at least one of a precedence constraint, a time window constraint, a priority constraint, or a conditional constraint); and

(d) a conditional constraint, wherein said conditional constraint limits scheduling of said tasks based on an occurrence of at least one event (See column 3, lines 20-40, column 4, lines 20-40, column 6, lines 1-10, which includes at least one of a precedence constraint, a time window constraint, a priority constraint, or a conditional constraint).

9. As per claim 5, Syswerda discloses a scheduling system wherein the brute force module includes a task list and a schedule permutation generator and Syswerda discloses a schedule evaluator (See figure 5, column 2, lines 39-40, and column 5, lines 62-67, wherein tasks are listed in random order using no intelligence or reasoning. See at least figure 5, column 3, lines 40-49, column 4, lines 36-50, and column 6, lines 1-13, wherein evaluating the schedules

produced is disclosed). However, Syswerda does not expressly disclose a schedule evaluator in the brute force module.

Syswerda discloses a brute force module that generates random schedules of tasks from the task list as well as evaluating these randomly generated schedules through the application of constraints and evaluations. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to include the evaluation functionality in the brute force module in order to increase the efficiency of the system by evaluating the generated schedules at each stage of the process, thereby removing excess schedules and allowing fewer and better schedules to proceed to the next process.

10. As per claim 6, Syswerda discloses a scheduling system wherein said schedule permutation generator includes a first algorithm, said first algorithm enumerating each possible permutation of the time dependent tasks in a schedule (See column 5, lines 62-67, wherein the permutation generator creates schedules with complete lists including all the tasks).

11. As per claim 7, Syswerda teaches a scheduling system wherein said schedule evaluator is an efficiency evaluator (See at least figure 5, column 3, lines 20-30 and 40-49, column 4, lines 36-50, and column 6, lines 1-13, wherein the schedule evaluator is discussed that evaluates the ability of the system to produce a desired effect).

12. As per claim 8, Syswerda discloses a scheduling system wherein said efficiency evaluator includes at least one parameter selected from the group consisting of a time parameter, a cost parameter, and a user-defined parameter (See at least figure 5, column 3, lines 20-30 and 40-49, column 4, lines 36-50, and column 6, lines 1-13, wherein the evaluator evaluates on a parameter including at least one of time, cost, or user-defined).

13. As per claim 9, Syswerda teaches a scheduling system wherein said efficiency evaluator includes a second algorithm, said second algorithm determining an efficiency of each task based on a position of said task in a schedule (See column 5, lines 62-67, and column 6, lines 1-13, wherein each task of the randomly generated schedule is ordered and placed in its first legal position based on the constraints and then the schedule is evaluated based on the ordering).

14. As per claim 10, Syswerda discloses a scheduling system wherein said deterministic module includes a task list and a schedule permutation generator and Syswerda also discloses a schedule evaluator (See figures 5 and 7, column 2, lines 40-45, column 3, lines 20-40, column 4, lines 20-36, column 5, lines 62-67, and column 6, lines 1-13, and column 8, lines 1-7, wherein a deterministic module is disclosed that includes a task list and a schedule permutation generator. See at least figure 5, column 3, lines 40-49, column 4, lines 36-50, and column 6, lines 1-13, wherein evaluating the schedules produced is disclosed). However, Syswerda does not expressly disclose a dynamic programming module or a schedule evaluator in the module.

Syswerda discloses a deterministically programmed schedule builder that obtains a list of tasks and orders them using hard and soft constraints. Syswerda et al. also discloses when discussing the deterministically programmed schedule builder that this module can include a more complex scheduling techniques, as stated in column 6, lines 5-11, and column 7, lines 29-37. Dynamic programming considers changeable competing or conflicting forces, such as resource and time constraints, to produce multiple possible outcomes and therefore would be considered a specific type of deterministic scheduling by one of ordinary skill in the art. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to use dynamic programming techniques in the deterministic module of Syswerda in

order to increase the ability of the system to build better schedules while still acting deterministically, as stated in column 7, lines 29-40.

Furthermore, Syswerda discloses a module that generates schedules of tasks from the task lists as well as evaluating these generated schedules through the application of constraints and evaluations. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to include the evaluation functionality in the module in order to increase the efficiency of the system by combining the features, thereby removing excess schedules and allowing fewer and better schedules to proceed in a more timely manner.

15. As per claim 11, Syswerda teaches a scheduling system wherein said schedule permutation generator includes a third algorithm, said third algorithm:

(a) designating a number of tasks to store in memory at a given time (See column 3, lines 13-30, column 4, lines 20-32, and column 5, lines 51-61, wherein a number of tasks are stored in memory at a given time);

(b) placing the number of tasks in an order of efficiency (See column 4, lines 20-50, and column 6, lines 1-15, wherein the tasks are placed in the order of efficiency); and

(c) determining whether a task outside of the number of tasks is more efficient than a task within the number of tasks (See column 4, lines 20-50 and 60-67, column 5, lines 1-5, and column 6, lines 1-15 and 30-45, wherein the schedules are scored by weighting if a task outside the schedule versus inside the schedule).

16. As per claims 12 and 13, claims 12 and 13 are rejected using the same art and rationale as claims 7 and 8, respectively.

17. As per claim 14, Syswerda teaches a scheduling system wherein said efficiency evaluator includes a fourth algorithm, said fourth enumerating each possible permutation of time dependent tasks (See figures 5 and 7, column 2, lines 40-45, column 3, lines 20-40, column 4, lines 20-36, column 6, lines 1-10, and column 8, lines 1-7, that considers the constraints to determine an ordering of the tasks and tries ever possible permutation of the schedule using the constraints).

18. As per claim 15, Syswerda teaches a scheduling system wherein said genetic module includes a task list, a genetic schedule permutation generator, and a schedule evaluator (See figure 5, column 3, lines 45-67, column 4, lines 35-67, column 6, lines 32-50, wherein the module gets a task list, a permutation generator, and a portion that evaluates the schedule and eliminates low scoring members).

19. As per claim 16, Syswerda teaches a scheduling system wherein said genetic schedule permutation generator includes a fifth algorithm, said fifth algorithm mating a first schedule and a second schedule to breed a third schedule (See column 5, lines 25-35, which discuss crossovers).

20. As per claims 17, 18, and 19, claims 17, 18, and 19 are rejected using the same art and rationale as claims 7, 8, and 14, respectively.

21. As per claim 20, Syswerda teaches a scheduling system wherein said partitioner module includes a brute force partitioner and a residual evaluator (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36, column 5, lines 62-67, column 6, lines 1-10, and 14-35, column 8, lines 1-7, wherein tasks are listed in random order using no intelligence or reasoning. A partitioner gives the brute force module the ability to do the

scheduling and generate lists of the tasks. See also column 4, lines 20-50 and 60-67, column 5, lines 1-5, and column 6, lines 1-15 and 30-45, wherein the schedules are scored by weighting if a task outside the schedule versus inside the schedule).

22. Claims 21-26 and 28-54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Syswerda (U.S. 5,319,781) in view of Oba et al. (U.S. 5,241,465).

23. As per claim 21, Syswerda discloses a scheduling system with a brute force module with a brute force partitioner (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36, column 5, lines 62-67, column 6, lines 1-10, and 14-35, column 8, lines 1-7, wherein tasks are listed in random order using no intelligence or reasoning. A partitioner gives the brute force module the ability to do the scheduling and generate lists of the tasks). However, Syswerda does not expressly disclose a brute force time completion estimator.

Oba et al. discloses a time completion estimator (See column 2, lines 45-55, and column 4, lines 1-7 and 29-35, wherein the scheduling system has a time completion estimator that estimates the time allowable for scheduling to occur).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules to generate optimal schedules based on the constraints. Both Syswerda and Oba et al. also teach running the schedule optimization a number of times at which point the best schedule is chosen. See column 6, lines 32-45, of Syswerda which discusses running the optimization a predetermined number of times and see column 2, lines 45-55, and column 4, lines 1-7 and 29-35, of Oba et al. which discusses running the optimization a predetermined amount of time. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the

invention to include a time completion estimator that limited the amount of time for optimization in the system of Syswerda in order to increase the timeliness of the system in generating a schedule for a user by limiting the optimization and evaluation time to a specific interval at which point the best schedule generated thus far is chosen.

24. As per claim 22, Syswerda teaches a scheduling system wherein said brute force time completion estimator includes a seventh algorithm, said seventh algorithm:

(a) generating a given number of schedule permutations using the brute force module (See figure 5, column 2, lines 39-40, and column 5, lines 62-67, wherein tasks are listed in random order in schedule permutations using no intelligence or reasoning);

However, Syswerda does not expressly disclose steps (b) and (c).

Oba et al. discloses :

(b) estimating a time to complete all possible schedule permutations using the module based on the time taken to generate the given number of schedule permutations (See column 2, lines 45-55, and column 4, lines 1-12 and 29-45, wherein a time is estimated and stored); and

(c) determining whether the estimated time to complete all possible schedule permutations using the module exceeds a given time limit (See column 2, lines 45-55, and column 4, lines 1-12 and 29-45, wherein the estimated time is evaluated).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules to generate optimal schedules based on the constraints. Both Syswerda and Oba et al. also teach running the schedule optimization a number of times at which point the best schedule is chosen. See column 6, lines 32-45, of Syswerda which discusses running the optimization a predetermined number of times and see column 2, lines 45-55, and column 4, lines 1-7 and 29-

35, of Oba et al. which discusses running the optimization a predetermined amount of time. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to include a time completion estimator that estimates the time needed for schedule optimization and makes determinations concerning said estimated time in the system of Syswerda in order to increase the timeliness of the system in generating a schedule for a user by limiting the optimization and evaluation time to a specific interval at which point the best schedule generated thus far is chosen.

25. As per claim 23, Syswerda teaches a scheduling system with a deterministic module, a genetic module, and a residual evaluator (See figures 5 and 7, column 2, lines 40-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36 and 50-67, column 6, lines 1-10 and 14-35, and column 8, lines 1-7, that considers the constraints to determine an ordering of the tasks and a genetic algorithm being implemented. See also column 4, lines 20-50 and 60-67, column 5, lines 1-5, and column 6, lines 1-15 and 30-45, wherein the schedules are scored by weighting if a task outside the schedule versus inside the schedule). However, Syswerda does not expressly disclose a dynamic program module or a time completion estimator in the residual evaluator.

Oba et al. discloses a time completion estimator (See column 2, lines 45-55, and column 4, lines 1-7 and 29-35, wherein the scheduling system has a time completion estimator that estimates the time allowable for scheduling to occur).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules to produce optimal schedules based on the constraints. Both Syswerda and Oba et al. also teach running the schedule optimization a number of times at which point the best schedule is chosen. See column 6, lines 32-45, of Syswerda which discusses running the optimization a

predetermined number of times and see column 2, lines 45-55, and column 4, lines 1-7 and 29-35, of Oba et al. which discusses running the optimization a predetermined amount of time. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to include a time completion estimator that limited the amount of time for optimization in the system of Syswerda in order to increase the timeliness of the system in generating a schedule for a user by limiting the optimization and evaluation time to a specific interval at which point the best schedule generated thus far is chosen.

Furthermore, Syswerda discloses a deterministically programmed schedule builder that obtains a list of tasks and orders them using hard and soft constraints. Syswerda et al. also discloses when discussing the deterministically programmed schedule builder that this module can include a more complex scheduling techniques, as stated in column 6, lines 5-11, and column 7, lines 29-37. Dynamic programming considers changeable competing or conflicting forces, such as resource and time constraints, to produce multiple possible outcomes and therefore would be considered a specific type of deterministic scheduling by one of ordinary skill in the art. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to use dynamic programming techniques in the deterministic module of Syswerda in order to increase the ability of the system to build better schedules while still acting deterministically, as stated in column 7, lines 29-40.

26. As per claim 24, Syswerda teaches a scheduling system including a resource evaluator (See column 4, lines 10-46, which discloses scheduling based on resource constraints and checking to see if the schedule fulfills these constraints). However, Syswerda does not expressly disclose that the resource is hardware.

Oba et al. disclose that the resources being scheduled are hardware (See column 3, lines 55-65, which discloses device resources).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules including the use of resource to produce optimal schedules based on the constraints. It would have been obvious to one of ordinary skill in the art at the time of the invention to include hardware as a resource constraining the scheduling of Syswerda in order to increase the efficiency of the system in producing a schedule for a user by including all resources that would be used by said user in the course of the tasks.

27. As per claim 25, Syswerda discloses a scheduling system wherein said resource evaluator includes a memory module and a central processing unit (See column 4, lines 10-46, and the appendix, which discloses a resource evaluator implemented using a computer system). However, Syswerda does not expressly disclose that the resource is hardware.

Oba et al. disclose that the resources being scheduled are hardware (See column 3, lines 55-65, which discloses device resources).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules including the use of resource to produce optimal schedules based on the constraints. It would have been obvious to one of ordinary skill in the art at the time of the invention to include hardware as a resource constraining the scheduling of Syswerda in order to increase the efficiency of the system in producing a schedule for a user by including all resources that would be used by said user in the course of the tasks.

28. As per claim 26, Syswerda discloses a scheduling system wherein said residual evaluator includes an eighth algorithm, said eighth algorithm:

(a) generating a given number of schedule permutations using the deterministic module (See figures 5 and 7, column 2, lines 40-45, column 3, lines 20-40, column 4, lines 20-36, column 6, lines 1-10, and column 8, lines 1-7, that considers the constraints to determine an ordering of the tasks, which discloses generating a number of schedule combinations).

However, Syswerda does not expressly disclose a dynamic programming module or steps (b) and (c).

Oba et al. discloses :

(b) estimating a time to complete all possible schedule permutations using the module based on the time taken to generate the given number of schedule permutations (See column 2, lines 45-55, and column 4, lines 1-12 and 29-45, wherein a time is estimated and stored); and

(c) determining whether the estimated time to complete all possible schedule permutations using the module exceeds a given time limit (See column 2, lines 45-55, and column 4, lines 1-12 and 29-45, wherein the estimated time is evaluated).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules to generate optimal schedules based on the constraints. Both Syswerda and Oba et al. also teach running the schedule optimization a number of times at which point the best schedule is chosen. See column 6, lines 32-45, of Syswerda which discusses running the optimization a predetermined number of times and see column 2, lines 45-55, and column 4, lines 1-7 and 29-35, of Oba et al. which discusses running the optimization a predetermined amount of time. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to include a time completion estimator that estimates the time needed for schedule optimization and makes determinations concerning said estimated time in the system of

Syswerda in order to increase the timeliness of the system in generating a schedule for a user by limiting the optimization and evaluation time to a specific interval at which point the best schedule generated thus far is chosen.

Furthermore, Syswerda discloses a deterministically programmed schedule builder that obtains a list of tasks and orders them using hard and soft constraints. Syswerda et al. also discloses when discussing the deterministically programmed schedule builder that this module can include a more complex scheduling techniques, as stated in column 6, lines 5-11, and column 7, lines 29-37. Dynamic programming considers changeable competing or conflicting forces, such as resource and time constraints, to produce multiple possible outcomes and therefore would be considered a specific type of deterministic scheduling by one of ordinary skill in the art. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to use dynamic programming techniques in the deterministic module of Syswerda in order to increase the ability of the system to build better schedules while still acting deterministically, as stated in column 7, lines 29-40.

29. As per claim 28, Syswerda teaches a scheduling system for scheduling a plurality of time-dependent tasks, said scheduling system comprising:

(a) an enumerative brute force module (See figure 5, column 2, lines 39-40, and column 5, lines 62-67, wherein tasks are listed in random order using no intelligence or reasoning);

(b) a deterministic module (See figures 5 and 7, column 2, lines 40-45, column 3, lines 20-40, column 4, lines 20-36, column 6, lines 1-10, and column 8, lines 1-7, that considers the constraints to determine an ordering of the tasks);

(c) a genetic module (See figure 5, column 3, lines 50-67, column 4, lines 50-67, column 6, lines 14-35, wherein a genetic algorithm is implemented);

(d) a partitioner module for selecting one of said brute force module, said dynamic programming module, or said genetic module to generate a schedule for selecting said plurality of time-dependent tasks (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36, column 5, lines 62-67, column 6, lines 1-10 and 14-35, column 8, lines 1-7, and the appendix, which discloses a computer implementation which gives a portion ability to divide the scheduling responsibilities by selecting one of the modules. This portion has control over the flow of the schedule generations); and

(e) a constraint file, said constraints file providing an input to said partitioner module (See column 3, lines 20-40, and appendix, which discloses storing constraints in the computer implemented scheduling system).

(f) the functionality of grouping schedules by score and deleting those schedules grouped at specific score thresholds (See column 4, lines 20-32 and 60-67, column 5, lines 1-5, and column 6, lines 14-40, wherein schedules are evaluated, grouped by score, and deleted at a certain threshold score. Since duplicate schedules would attain the same score when evaluated by the system, duplicates would be grouped together and duplicates would be deleted).

However, Syswerda does not expressly disclose a dynamic programming module, said dynamic programming module including a hashing function having a low probability of collisions and a height-balanced binary tree for providing search insertion and deletion operations.

Oba et al. discloses a dynamic programming module with a height-balanced binary tree for providing search insertion and deletion operations (See at least figure 10, column 1, lines 62-67, column 2, lines 1-5, column 4, lines 1-13, column 5, lines 10-34 and 48-65, and column 8, lines 32-65, wherein a height-balanced binary tree provides for search insertion and deletion operations for the dynamic scheduling tool).

However, Oba et al. does not expressly disclose hashing function having a low probability of collisions.

Both Syswerda and Oba et al. teach scheduling systems that generate, evaluate, and eliminate schedules to generate optimal schedules based on the constraints. Syswerda et al. also discloses when discussing the deterministically programmed schedule builder that this module can include a more complex scheduling techniques, as stated in column 6, lines 5-11, and column 7, lines 29-37. Dynamic programming considers changeable competing or conflicting forces, such as resource and time constraints, to produce multiple possible outcomes and therefore would be considered a specific type of deterministic scheduling by one of ordinary skill in the art. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to use dynamic programming techniques in the deterministic module of Syswerda in order to increase the ability of the system to build better schedules while still acting deterministically, as stated in column 7, lines 29-40.

Furthermore, Syswerda teaches grouping schedules by score and deleting those schedules grouped at specific score thresholds as well as deterministically building schedules by applying soft and hard constraints to lists of tasks. Hashing functions are well known in the computer programming art and are used to identify and map values to a location for faster data retrieval. It

is also known in the art of programming that a hash function should be fast and it should cause as few collisions as possible so that it does not produce the same value from two different inputs. It would have been obvious to one of ordinary skill in the art at the time of the invention to use a hashing function to group the schedules in order to more efficiently map schedules of duplicate task lists and/or duplicate scores to the same location for easy deletion.

30. As per claims 29-52, claims 29-52 recite system claims containing comparable elements to the system of claims 3-26, respectively. Therefore, claims 29-51 are rejected using the same art and rationale as the rejections of claims 3-26, respectively.

31. As per claim 53, Syswerda teaches a method for scheduling a plurality of time-dependent tasks, said method comprising the steps of:

(a) providing an input to a partitioner module (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36, column 5, lines 62-67, column 6, lines 1-10 and 14-35, column 8, lines 1-7, and the appendix,);

(b) selecting a scheduling module for the group consisting of: an enumerative brute force module, a deterministic module, a genetic module (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36 and 50-67, column 5, lines 62-67, column 6, lines 1-10 and 14-35, and column 8, lines 1-7, wherein scheduling modules exist to which the schedule task lists are sent); and

(c) generating a schedule (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36 and 50-67, column 5, lines 62-67, column 6, lines 1-10 and 14-35, and column 8, lines 1-7, wherein a schedule is generated using one of the modules).

However, Syswerda does not expressly disclose a dynamic programming module that includes a hashing function having a low probability of collision.

Syswerda discloses a deterministically programmed schedule builder that obtains a list of tasks and orders them using hard and soft constraints. Syswerda et al. also discloses when discussing the deterministically programmed schedule builder that this module can include a more complex scheduling techniques, as stated in column 6, lines 5-11, and column 7, lines 29-37. Dynamic programming considers changeable competing or conflicting forces, such as resource and time constraints, to produce multiple possible outcomes and therefore would be considered a specific type of deterministic scheduling by one of ordinary skill in the art. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to use dynamic programming techniques in the deterministic module of Syswerda in order to increase the ability of the system to build better schedules while still acting deterministically, as stated in column 7, lines 29-40.

Furthermore, Syswerda teaches grouping schedules by score and deleting those schedules grouped at specific score thresholds as well as deterministically building schedules by applying soft and hard constraints to lists of tasks. Hashing functions are well known in the computer programming art and are used to identify and map values to a location for faster data retrieval. It is also known in the art of programming that a hash function should be fast and it should cause as few collisions as possible so that it does not produce the same value from two different inputs. It would have been obvious to one of ordinary skill in the art at the time of the invention to use a hashing function to group the schedules in order to more efficiently map schedules of duplicate task lists and/or duplicate scores to the same location for easy deletion.

32. As per claim 54, Syswerda discloses a method for scheduling a plurality of time-dependent tasks, said method comprising the steps of:

(a) providing an input to a partioner module (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36, column 5, lines 62-67, column 6, lines 1-10 and 14-35, column 8, lines 1-7, and the appendix,);

(b) selecting a scheduling module for the group consisting of: an enumerative brute force module, a deterministic module, and a genetic module (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36 and 50-67, column 5, lines 62-67, column 6, lines 1-10 and 14-35, and column 8, lines 1-7, wherein scheduling modules exist to which the schedule task lists are sent); and

(c) generating a schedule (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36 and 50-67, column 5, lines 62-67, column 6, lines 1-10 and 14-35, and column 8, lines 1-7, wherein a schedule is generated using one of the modules).

(d) the functionality of grouping schedules by score and deleting those schedules grouped at specific score thresholds (See column 4, lines 20-32 and 60-67, column 5, lines 1-5, and column 6, lines 14-40, wherein schedules are evaluated, grouped by score, and deleted at a certain threshold score. Since duplicate schedules would attain the same score when evaluated by the system, duplicates would be grouped together and duplicates would be deleted).

However, Syswerda does not expressly disclose a dynamic programming module, a hashing function having a low probability of collision, or a height-balanced binary tree for providing search insertion and deletion operations.

Oba et al. teaches a height-balanced binary tree for providing search insertion and deletion operations (See at least figure 10, column 1, lines 62-67, column 2, lines 1-5, column 4, lines 1-13, column 5, lines 10-34 and 48-65, and column 8, lines 32-65, wherein a height-balanced binary tree provides for search insertion and deletion operations).

However, Oba et al. does not expressly disclose a hashing function.

Both Syswerda and Oba et al. teach scheduling systems that generate, evaluate, and eliminate schedules to generate optimal schedules based on the constraints. Specifically, Syswerda teaches grouping schedules by score and deleting those schedules grouped at specific score thresholds as well as deterministically building schedules by applying soft and hard constraints to lists of tasks. Hashing functions are well known in the computer programming art and are used to identify and map values to a location for faster data retrieval. It is also known in the art of programming that a hash function should be fast and it should cause as few collisions as possible so that it does not produce the same value from two different inputs. It would have been obvious to one of ordinary skill in the art at the time of the invention to use a hashing function to group the schedules in order to more efficiently map schedules of duplicate task lists and/or duplicate scores to the same location for easy deletion.

Furthermore, Syswerda discloses a deterministically programmed schedule builder that obtains a list of tasks and orders them using hard and soft constraints. Syswerda et al. also discloses when discussing the deterministically programmed schedule builder that this module can include a more complex scheduling techniques, as stated in column 6, lines 5-11, and column 7, lines 29-37. Dynamic programming considers changeable competing or conflicting forces, such as resource and time constraints, to produce multiple possible outcomes and therefore

would be considered a specific type of deterministic scheduling by one of ordinary skill in the art. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to use dynamic programming techniques in the deterministic module of Syswerda in order to increase the ability of the system to build better schedules while still acting deterministically, as stated in column 7, lines 29-40.

33. Claim 27 is rejected under 35 U.S.C. 103(a) as being unpatentable over Oba et al. (U.S. 5,241,465) in view Syswerda (U.S. 5,319,781).

34. As per claim 27, Oba et al. discloses a dynamic programming module adapted to provide at least one solution for a scheduling problem, said dynamic programming module including:

(a) a dynamic program module (See column 2, lines 5-25, wherein the schedule is dynamically generated using a strategy); and

(b) a height-balanced binary tree for providing search insertion and deletion operations (See at least figure 10, column 1, lines 62-67, column 2, lines 1-5, column 4, lines 1-13, column 5, lines 10-34 and 48-65, and column 8, lines 32-65, wherein a height-balanced binary tree provides for search insertion and deletion operations).

However, Oba et al. does not expressly disclose hashing function, said hashing function being capable of detecting duplicate solutions generated.

Syswerda discloses the functionality of grouping schedules by score and deleting those schedules grouped at specific score thresholds (See column 4, lines 20-32 and 60-67, column 5, lines 1-5, and column 6, lines 14-40, wherein schedules are evaluated, grouped by score, and deleted at a certain threshold score. Since duplicate schedules would attain the same score when

evaluated by the system, duplicates would be grouped together and duplicates would be deleted). However, Syswerda does not specifically disclose a hashing function.

Both Syswerda and Oba et al. teach scheduling systems that generate, evaluate, and eliminate schedules to generate optimal schedules based on the constraints. Specifically, Syswerda teaches grouping schedules by score and deleting those schedules grouped at specific score thresholds. Since hashing functions are well known in the computer programming art and are used to identify and map values to a location, it would have been obvious to one of ordinary skill in the art at the time of the invention to use a hashing function to group the schedules in order to more efficiently map schedules of duplicate task lists and/or duplicate scores to the same location for easy deletion.

Response to Arguments

35. Applicant's arguments with regards to the rejections based on Oba et al. (U.S. 5,241,465) and Syswerda (U.S. 5,319,781) have been fully considered, but they are not persuasive. In the remarks, Applicant argues that (1) Syswerda does not disclose a dynamic programming module that includes a hashing function having a low probability of collision and (2) that while hashing functions are per se known in the art, there is no suggestion in the prior art that would have led one of ordinary skill in the art to apply hashing functions.

In response to argument (1) of the Applicant, Examiner agrees with Applicant that Syswerda does not disclose a dynamic programming module that includes a hashing function having a low probability of collision, as stated above in the 35 USC § 103 rejections. However, Examiner maintains that it would have been obvious to one of ordinary skill in the art at the time of the invention to combine a dynamic programming module with a hashing function with the

teachings of Syswerda, as set forth in the rejections above. Since Syswerda does disclose a deterministically programmed schedule builder that obtains a list of tasks and orders them using hard and soft constraint and that the schedule builder can include a more complex scheduling technique (column 6, lines 5-11, and column 7, lines 29-37), it would have been obvious to include dynamic programming since it would be considered a specific type of deterministic scheduling by one of ordinary skill in the art.

As for the claimed hashing function, Examiner points out that there is no specific recitation in the claims of how this hashing function specifically applies to the claimed scheduling beyond using the hashing function that is known in the art of programming. Low probability of collisions is a known and desired trait of hashing function. Since Syswerda teaches grouping schedules by score and deleting those schedules grouped at specific score thresholds as well as deterministically building schedules by applying soft and hard constraints to lists of tasks and since hashing functions are well known in the computer programming art and are used to identify and map values to a location for faster data retrieval, it would have been obvious to one of ordinary skill in the art at the time of the invention to use a hashing function to group the schedules in order to more efficiently map schedules of duplicate task lists and/or duplicate scores to the same location for easy deletion.

Examiner further points out that the dynamic programming module with the hashing function may not ever even be applied by the claimed scheduling system since a partitioner module only selects one of three modules when performing scheduling.

In response to argument (2) of the Applicant, Examiner reminds the Applicant that obviousness can only be established by combining or modifying the teachings of the prior art to

produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988)and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). As discussed above, the claims contain no specific recitation of how this hashing function specifically applies to the claimed scheduling beyond using the hashing function that is known in the art of programming. Therefore, one of ordinary skill in the art would have been motivated to use a hashing function in order to gain the known benefits of hash functions, such as speed of data retrieval. Examiner suggests that if something more specific is meant by the hashing function, the claims should be amended to reflect this specificity.

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Nemes (U.S. 5,121,495) teaches using a hashing technique to store and retrieve information and scheduling using this information.

Nemes (U.S. 5,893,120) teaches collision resolution with hashing algorithms.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Beth Van Doren whose telephone number is (703) 305-3882. The examiner can normally be reached on M-F, 8:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tariq Hafiz can be reached on (703) 305-9643. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

bvd
April 5, 2004

Susanna Diaz
Susanna Diaz
Primary Examiner
AU 3623